Understanding Strategy Differences in a Fault-Finding Task

Maik B. Friedrich[1] <maik.friedrich@dlr.de>

Frank E. Ritter[2] <frank.ritter@psu.edu>

27 jun 2019

10270 words (body)

1- German Aerospace Center, Braunschweig

2- College of IST, Penn State

Corresponding author:

Maik Friedrich, Maik.Friedrich@dlr.de

+49 (0) 531 295-2598

German Aerospace Center

Lilienthalplatz 7

Braunschweig, NI D-38108

Abstract

This article examines strategy choices for how people find faults in a simple device by using models of several strategies and new data. Diag, a model solving this task, used a single strategy that predicted the behavior of most participants in a previous study with remarkable accuracy. This article explores additional strategies used in this reasoning task that arise when less directive instructions are provided. Based on our observations, five new strategies for the task were identified and described by being modeled. These different strategies, realized in different models, predict the speed of solution while the participant is learning the task, and were validated by comparing their predictions to the observations (r^2= .27 to .90). The results suggest that participants not only created different strategies for this simple fault-finding task but that some also, with practice, shifted between strategies. This research provides insights into how strategies are an important aspect of the variability in learning, illustrates the transfer of learning on a problem-by-problem level, and shows that the noisiness that most learning curves show can arise from differential transfer between problems.

I. Introduction

Is it possible to construct a general learning model that can capture individual differences in strategies as these strategies develop? Can we find predictable sources of variability in learning, across time and across learners? Many believe the answers to these questions are yes, but modeling human behavior using different strategies and comparing them with the time course of individual learning behavior remains difficult. There is little research on different strategies for a task, because to do so tasks that are complex enough to allow multiple strategies have to be used and individual behavior must be traced and analyzed.

Siegler (1988b) and Elio and Scharf (1990) have argued that we should not average over strategies, and that understanding individual differences is important. Averaging over strategies will distort results, and hide important and reliable effects. So, in this article, we report how we modeled multiple strategies while they were learned, how they matched some subjects' performance fairly well, and how they inform us about behavior, transfer, and learning. We will also see that there are yet more strategies subjects use that we cannot yet derive.

We examine strategy use by analyzing a diagrammatic reasoning task with human data and cognitive models. Measurable improvement begins when the same task is performed a second time with the information learned from the first time. This typically leads to greater efficiency. Performance is improved with practice. For some tasks, improvement continues for over 100,000 trials (Seibel, 1963). John and Lallement (1997) and Delaney, Reder, Staszewski, and Ritter (1998) demonstrated that improvements could be a result of strategy development and change.

Strategy analysis is an important step for modeling a task using a cognitive architecture. Kieras and Meyer (2000) described the cognitive modeling practice as the process of making an intuitive guess about the strategy and comparing the predicted data with human data. If the

predictions and data fit, the model and strategy are validated, if not, the researcher continues working.

A model that learns helps describe how humans use knowledge to solve problems and how the repetitive application of the same knowledge leads to faster performance. A good overview of the categories of human learning is provided by Ashby and Maddox (2005). There are several learning models implemented in cognitive architectures. These models include procedural knowledge compilation (Anderson, 2007; Gonzalez, Lerch, & Lebiere, 2003; Lebiere, Wallach, & West, 2000; Pavlik, 2007) and base level learning (Altmann & Trafton, 2002) in ACT-R, connection strengthening in PDP models (e.g.,O'Reilly & Munakata, 2000), rule creation from impasses using analogical reasoning (VanLehn & Jones, 1993), constraint elaborations (Ohlsson, 2007), and both declarative and procedural learning mechanisms (e.g., Feigenbaum & Simon, 1984; Gobet & Lane, 2007; Könik et al., 2009; Rosenbloom, Laird, & Newell, 1987). These models had their predictions compared to averaged data, but have not had their predictions compared to individual learners.

A problem in the study of complex problem solving, especially in a learning context, is the spectrum of individual differences. Understanding individual differences can help understand the acquisition and application of complex skills (Card, Moran, & Newell, 1983; Kieras & Meyer, 2000). A way to study individual differences is to take an existing task and model of the task, and allow participants to solve the task in multiple ways. A previously reported study (Ritter & Bibby, 2008) used a model that learned and matched ($r^2 > .9$) most participants' learning across a series of problems with a single strategy. Readers and reviewers have suggested that the good fits may have been due to the task instructions being directive. In this report, we examine further strategies that can arise with less directive task instructions.

Previous models of individual differences

Modeling individual differences in problem solving has been done at least in three ways. The first and most straightforward way is to model individual differences in problem solving as differences in global system parameters. Card et al. (1983) proposed a model with three different sets of parameters: Fastman (parameters are set to deliver the fastest performance), Middleman (parameters are set to deliver a nominal performance), and Slowman (parameters are set to deliver the slowest performance). Daily, Lovett, and Reder (2001) varied working memory capacity to model individual differences. Kase, Ritter, Bennett, Klein, and Schoelles (2017) modeled individual differences in problem solving as changes in a set of parameters. These models do not include different strategies or learning over time, but represent the behavioral differences across subjects as different parameters in models.

The second, and more complex, way is to model individual differences in problem solving as differences in knowledge. This can be done by including either different knowledge sets or different representations. Kieras and Meyer (2000) proposed a Bracketing Heuristic where three strategies in the range of performance (slow, medium, fast) are modeled, and the range of human performance is predicted as the first step of cognitive task analysis. Best and Lovett (2006) developed a methodology for using a normative task model as the basis for training methodology and thereby gained insights into the differences between task performance of a rule-based model and an instance-based model. Anzai and Simon (1979) used different knowledge to model four different strategies in solving the Tower of Hanoi and explained how these changes could occur automatically. These models did not include learning within or between strategies, however.

The third way to model individual differences in problem solving is how they arise through learning. This is demonstrated in work by Siegler (1988a, 1988b). He used differences in strategies, their error rates, and learning, to model how children learned to perform arithmetic

using up to five different strategies. He modeled how children shifted between strategies through learning. (He has also argued that modeling individual strategies is important to understand sets of data (Siegler, 1987)). This approach for modeling learning has been implemented in cognitive architectures as well. Larkin's Able model (Larkin, 1981; Larkin, McDermott, Simon, & Simon, 1980) showed how novices use backward chaining of inference rules to solve physics problems, and how learning changes this process into forward chaining of inference rules. Able only modeled shifting between two strategies, but could be seen as having multiple strategy variants representing different amounts of the knowledge proceduralized.

Thus, previous work has argued for the need and the usefulness of representing strategies, particularly Simon and Reed (1976) and Siegler (1987). These models have shown that individual differences in problem solving arise in multiple tasks, and that modeling individual strategies can lead to a better understanding of how the task is performed and how learning strategies arise (also see Delaney et al., 1998; VanLehn, 1991). This work has also presented the case that strategies are important for understanding behavior, and that learning is important as a mechanism to shift between and to evolve strategies. Considering the previous literature, the influence of multiple strategies on performance and learning should be explored further.

II. The Fault Finding Task: Troubleshooting a faulty circuit

A task that can be solved with multiple strategies while learning can be used to examine performance and learning in detail. As learning modifies these strategies, we can also see the effects of learning on the strategies over time. Troubleshooting tasks have been used to study learning in the past. Lesgold, Lajoie, Bunzon, and Eggan (1992) developed a simple troubleshooting task. White and Frederiksen (1987) used a troubleshooting task to focus on the role of scientific explanation and its use in instructions. We used an electrical troubleshooting

task to study problem solving. However, troubleshooting tasks used in earlier studies concentrated on single strategies to solve the task (e.g.,Besnard, 2000; Besnard & Bastien-Toniazzo, 1999) and did not consider learning and its effect on strategies used.

We begin with a short introduction of the Fault Finding Task (Bibby & Payne, 1993). The setup consists of an interface (Fig. 1) that controls the circuit that is represented by a diagram shared with participants (Fig. 2). The interface consists of lights (at the top) showing if the components work and switches (at the bottom) showing via arrows the direction in which the circuit is routed. The system consists of a power source (PS), two energy boosters (EB1 and EB2), accumulators (MA, SA1, and SA2), laser bank (LB), wiring, switches, and indicator lights. Studies using this experimental setup have examined different tasks, for example, finding the wrong switch position, operating the device (Bibby & Payne, 1993), or the Fault-Finding Task (Ritter & Bibby, 2008). The Fault Finding Task, which we use here, has the goal of identifying the faulty component that constrains the energy on its way from PS to LB. We often refer to the task by the broken part to be found (e.g., the SA1 task).

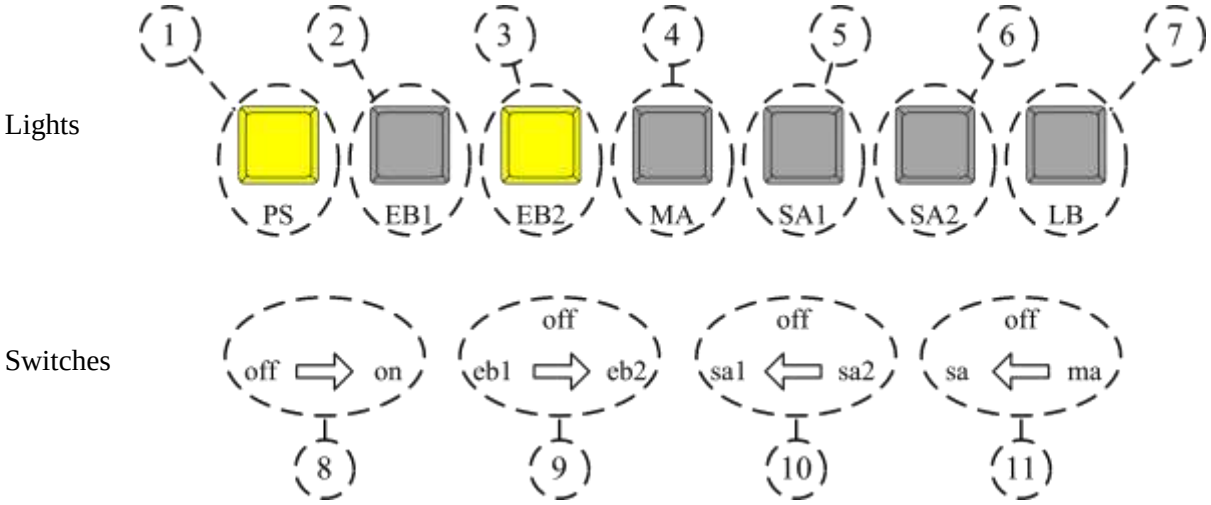


Figure 1. The Fault Finding Task interface (Ritter & Bibby, 2008). Darker lights are off (e.g., EB1), lighter lights are on (e.g., PS). The broken component represented in this figure is SA1. Dashed lines, numbers, and labels on the left are added here for later reference.

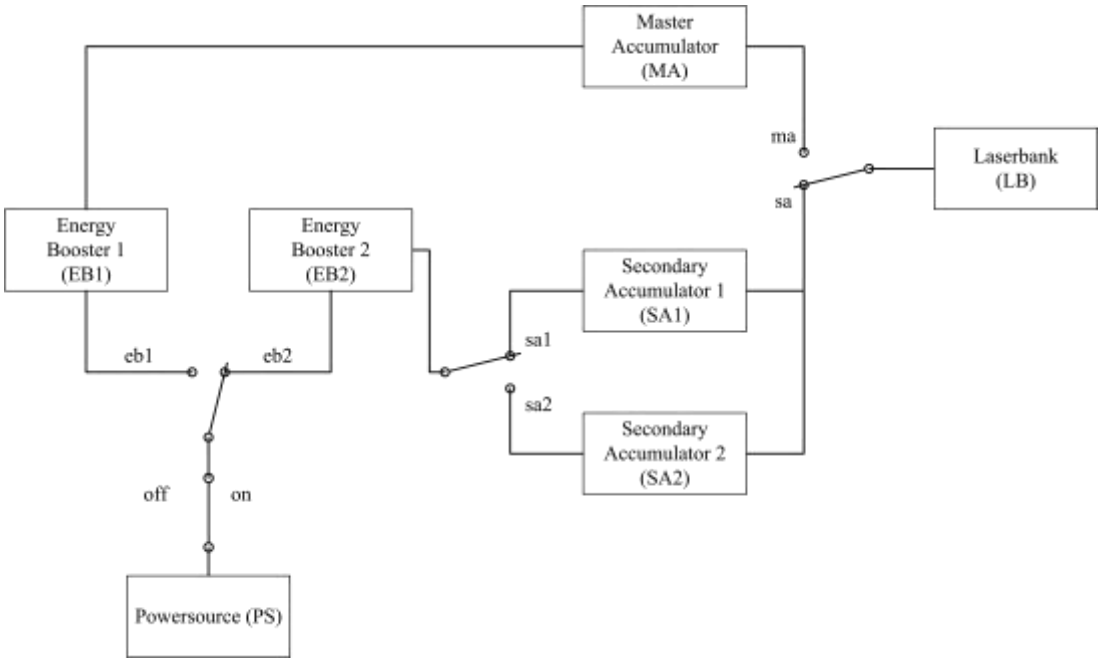


Figure 2. The Fault Finding Task circuit.

In the rest of the article, we will describe the model and new human data that appear to have been performed with a variety of strategies. Then, we will explore what the model and data tell us about learning.

Review of the single strategy model

The original Diag model solves the Fault Finding Task. It is built on the idea that "procedural, declarative and episodic learning all are taking place within individual problem-solving episodes" (Ritter & Bibby, 2008). Diag learns at a similar rate to most participants, predicting learning rates on a trial-by-trial basis, predicting transfer across a series of 20 tasks accurately for 8 out of the 10 participants in the previous study.

The Diag[1] model was implemented in Soar 6, an instantiation of Newell (1990) Unified Theory of Cognition. Soar is a symbolic cognitive architecture, initially developed by Laird, Newell, and Rosenbloom (1987). It provides a view of what cognition is and how it could be

[1] The Diag model, described in Ritter and Bibby (Ritter & Bibby, 2008), is available online at acs.ist.psu.edu/projects/diag.

implemented as a software architecture. Soar provides the user with the necessary tools to implement Newell's UTC. This implementation is realized implicitly by using rules. This rule-based concept makes finding the theoretical elements of Newell's UTC difficult when examining Soar code. This also leads to a low-level of abstraction and not to the explicitness that the theory proposes.

Soar is based on the Problem Space Computational Model (PSCM) (Lehman, Laird, & Rosenbloom, 1996; Newell, Yost, Laird, Rosenbloom, & Altmann, 1991). The PSCM is based on the primitive acts that are performed while using problem spaces to achieve a goal (Laird, 2012). Problem spaces are common collections of states, sets of states, goals, and a set of valid operators that contain the constraints under which to apply themselves. The top-level goal is to transform the initial state into the goal state. This goal is reached by applying operators. A state consists of a set of literals that describe the knowledge of the agent and the present model of the environment. The strategy of the Soar agent is defined as the movement through a problem space.

We define strategies as behaviors that differ in the order and selection of operators when solving a problem. Using this low-level approach, several possible strategies are identified that allow a detailed view of the differences between individual performances. Strategies lead to different performances that solve the same task (Meyer et al., 1995), and we will be able to examine those differences in detail here. Learning in this model is information or strategies that are acquired and used to change the problem-solving process.

On the PSCM level, Diag consists of seven problem spaces that are hierarchically ordered. This is shown in Fig. 3. Within these problem spaces, 20 operators are grouped to perform the problem-solving task. These problem spaces and operators are implemented through

170 Soar rules. These rules realize the problem space elements. Rules are a low-level representation and do not provide the explicit representation that the PSCM theory proposes.

The model can be described as having three types of memories and learning (Ritter & Bibby, 2008). Declarative memory describes acquired facts, for example, component A is connected to component B. Procedural memory stores action sequences, for example, how to play a game or how to decide which object you should look at next, or how to decide the current state of the object in the Diag interface task. Episodic knowledge is used by humans to remember events, for example, the result of an operation. As the models solve problems, they create knowledge from search in subspaces, using Soar's chunking mechanism, that helps implement more quickly the declarative memories about how components are connected, how to interpret the information from looking at the interface, and how to implement operators.

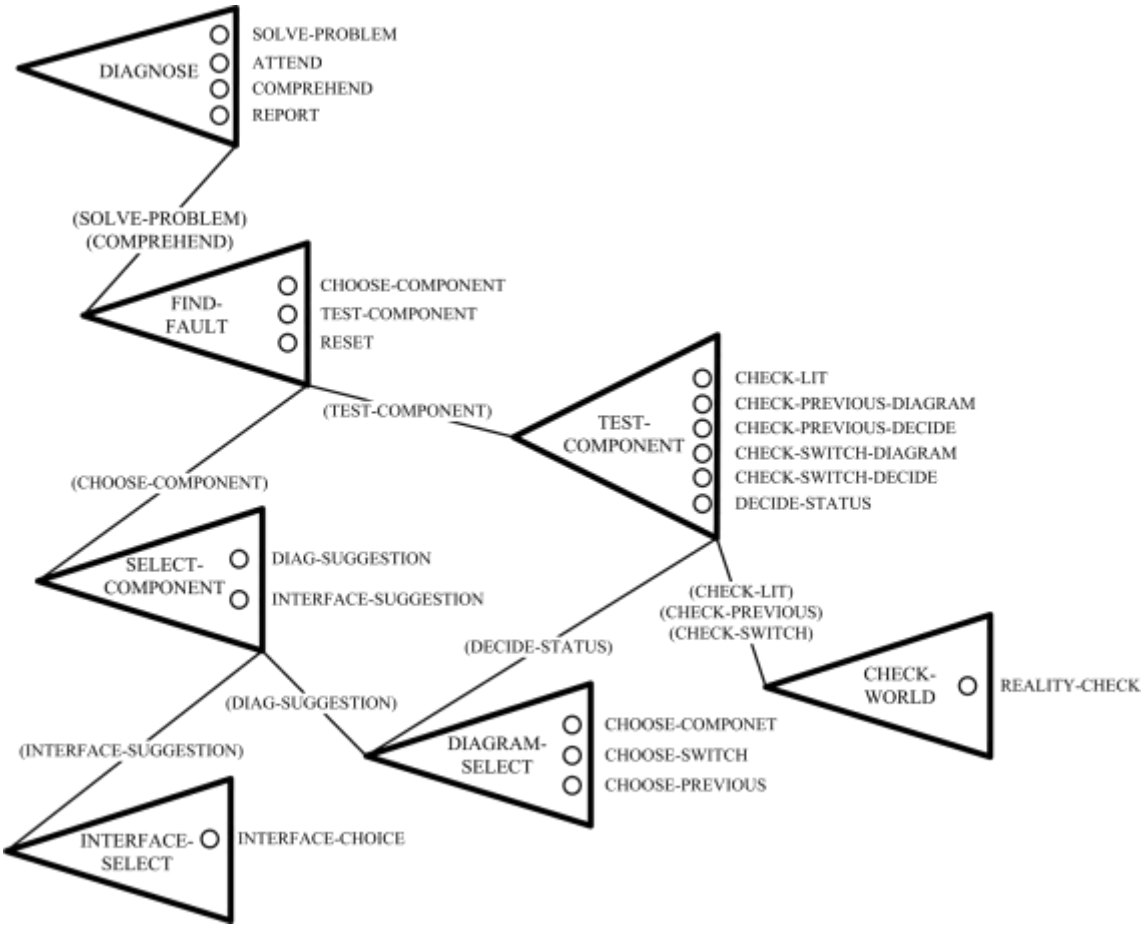


Figure 3. Problem space structure of the Diag Model. Operators are typically used from left to right, and when they cannot be implemented directly, by an associated problem space below them.

When solving the Fault-Finding Task, Diag identifies the faulty component by using the interface combined with knowledge about the diagram. The strategy that Diag uses to solve the Fault-Finding Task is based on lights, and uses the schematic and switch settings to find the fault. It works from left to right across the interface starting by checking the PS (Fig. 1 Component 1) component's light. If the light is off, PS is the broken part, otherwise Diag checks the EB1 light (Fig. 1 Component 2). If EB1 is on, the model moves to the next light. If EB1 is off, Diag checks the second switch (Fig. 1 Component 9) to see if the energy is routed through EB1. If the second switch is set to EB1, the broken part is identified, otherwise Diag moves down the other path defined by the schematic to a different component. This pattern of using the diagram

information and schematic information continues until the faulty component is found. There are other strategies as well (which we explore below), and each strategy has a different order in which information is gathered and used (Bibby & Payne, 1996; Meyer et al., 1995).

Diag and Diag-H (our reimplemented model) are both divided into several problem spaces to solve the Fault-Finding Task. The problem space DIAGNOSE[2] represent the top level and starts the problem solving. The sub-problem space of DIAGNOSE is FIND-FAULT that coordinates the search for the faulty component by selecting and testing the next component. FIND-FAULT has two sub-problem spaces, SELECT-COMPONENT and TEST-COMPONENT. SELECT-COMPONENT selects the next component either from the interface or the circuit schematic. TEST-COMPONENT then tests the selected component with three tests (light lit up or not; the switch is pointed to it; the previous component is unbroken). The TEST-COMPONENT problem space relies on the CHECK-WORLD problem space that gets information for the tests from the world, that is, the current state of the interface. With practice solving the Fault-Finding Task, knowledge about connections in the system is learned, as well as expectations about whether a light should be on given a configuration of switches.

Diag predictions were generated for Ritter and Bibby's 10 participants, who each experienced a unique stimulus set[3]. The instructions given to the participants are described below, because they are identical to the study conducted in this article. The number of (Soar) model cycles Diag needed to solve a series of problems was recorded for the sets of problems that the participants saw and solved. The predicted times to solve these sets of problems were regressed against the problem-solving times for each participant individually. The results showed that for all participants the average r^2 was 79%. The behavior of two participants did not fit with

[2] We put model operators and problem spaces in all caps to distinguish them.
[3] The stimulus sets can be found in Ritter and Bibby (2008) and the accompanying web site, acs.ist.psu.edu/projects/diag.

a reliable correlation the predictions of the Diag model. This could be explained if they used different strategies for doing this task. However, two participants were not enough for analyzing different Fault-Finding Task strategies. Therefore, additional data were needed. The work presented in this article focuses on examining alternative strategies for completing the Fault-Finding Task that was not modeled by Ritter and Bibby (2008).

Analyzing strategies that solve the Fault Finding Task requires a set of models with different strategy implementations. Therefore, a reimplemented Diag model in a high-level modeling language has considerable advantages because it is more easily modified and also multiple strategies can be implemented clearly. A high-level behavior representation language, Herbal (Cohen, Ritter, & Haynes, 2010), was used for the reimplementation. Herbal is a high-level language based on the PSCM, allowing for models to be created at a level of abstraction above the standard production level. Herbal allows the cognitive modeler to focus on the architectural aspects of their cognitive agent while the details of programming are managed by the Herbal compiler. This way Herbal represents a step towards the development of tools that support modelers of intelligent agents and cognitive behavior. Herbal is designed to create agents which have the ability to explain themselves. This is possible through the formalization of the programming process in Herbal with the help of an explicit ontology of classes that represent concepts of the PSCM. These classes are the basis for the Herbal structure (Cohen, Ritter, & Bhandarkar, 2007).

A Herbal implementation of Diag (Diag-H) creates a Soar model, but more importantly facilitates adaptations of the model (e.g., changing the strategies). Diag-H is more flexible, reusable and expandable than raw Soar code. Different strategies can be created relatively easily, as we will present in the analysis section. Herbal produces the resulting models as Soar models,

which then are modified through experience using the existing learning mechanisms in the Soar architecture.

Table 1 provides an example to illustrate the Herbal XML-based code and the equivalent in Soar code for a condition called CheckWorld-MA-On. This condition tests whether the MA (main accumulator) light is ON or OFF on the interface (Fig. 1 and Fig. 2). This translation is straightforward because Soar has clear support for the concept of conditions and both Soar and Herbal implement the PSCM.

Table 1. A Translation from a Herbal Condition to Soar (in style of Cohen et al. (2010))

Architecture	Source Code
Herbal XML Language	<condition name='CheckWorld-MA-On'> <match type='Input'> <restrict field='ma'> <eq>true</eq> </restrict> </match> </condition>
Compiled Soar Code	(<i2> ^|H-Diag.types.Input| <H-Diag-types-Input2>) (<H-Diag-types-Input2> ^ma <ma433> true)

Overview of the paper

In the remainder of the paper, we will present a study where the instructions allowed the participants to generate a wider range of strategies. We use the mouse movements from the participants' early trials to infer potential strategies. These strategies are then implemented in Soar, and the strategies are run on the series of tasks that the users also saw. The strategies generate a series of predicted task solution times and the times decrease with practice according to Soar's learning mechanism. By comparing the strategies' predicted response times to the human data, we will see which strategies are used. These results are then discussed.

III. Finding additional strategies

To identify additional strategies a new user study with the diagrammatic reasoning task was run with less directive instructions to the participants. The goal was to identify new strategies by less directive instructions and through a larger number of participants.

Participants

Thirty-seven undergraduate students from the Pennsylvania State University and the Otto Friedrich University in Bamberg (7 female and 30 male; between 18 and 29 years; 16 Americans and 21 Germans) performed the task. The participants were randomly assigned to 11 groups of three participants, except group 11, which consisted of seven participants. Every group was given a different stimulus set of Fault-Finding Task problems. Their majors were psychology (8), computer science (12), economics (5), business informatics (8), and mechanical engineering (4). The Americans were paid $7 for participating whereas the Germans participated out of interest. All participants had a basic understanding of electric circuits that made the introduction to the Fault-Finding Task relatively easy, but none of them were familiar with the task. Participant IDs (used later in this article) were issued according to the following examples: G14_10 stands for the 14th German (G) participant who solved stimulus set 10, while A1_3 stands for the first American (A) participant who solved stimulus set 3.

Materials and apparatus

This user study was based on the original study materials of Ritter and Bibby (2008). The materials used for this study were a general task introduction and the task environment. The introduction consisted of a background story to motivate the participants, supported by a picture of the control interface (Fig. 1) and a schematic of the underlying circuit (Fig. 2). No initial example of the task was provided in this study, in contrast to Ritter and Bibby (2008), to encourage different strategies. The instructions also contained a set of frequently asked questions

and their answers. These questions were collected from pilot participants and undergrad students from a human-computer interaction class at Penn State.

The task environment was presented with PDF files on a MacBook with a 15.4 inch screen and a resolution of 1920 by 1200. The task environment consisted of 48 slides (2 introduction slides, 5 training slides with no task information, 20 task slides and 20 pause slides between the task slides)[4]. The tasks were randomly ordered from a set of 20. Example series are included below in Fig. 6 and 7 and all sets in the online materials. Participants used the PAGE-DOWN key to switch to the next slide. The participants used a mouse to navigate on the task slides and select the faulty components. The participants' performance while solving a stimulus set was captured with the RUI mouse and keystroke logger (Kukreja, Stevenson, & Ritter, 2006). A Java application, DiagData-Analyser, was used to analyze the keystroke and mouse interactions, similar to MouseTracker (Freeman & Ambady, 2010).

Design and procedure

Each participant attended one experimental session that lasted approximately 30 minutes. At the beginning, after consent, every participant was asked to read the instruction materials. While studying the instructions, they were allowed to ask questions. The participants were told to pay special attention to the schematic representation of the circuit. After a five minute study period, the participants had to draw the schematic from memory. If they were unable to reproduce the schematic, an additional five-minute study period was given. No participant needed more than 10 minutes to reproduce the schematic accurately, and 28 of the participants succeeded in 5 minutes.

[4] This training environment and the task environment are available (http://acs.ist.psu.edu/code/diag-H/diag-H-stimuli/), and this data set and model are available as well (acs.ist.psu.edu/projects/diag).

After reproducing the schematic, the instructions were removed and the training phase began. The participants were instructed to put their regular mouse hand on the mouse and their other hand on the PAGE-DOWN key. The participants had five training slides for practicing clicking on the screen and changing the slides.

After training, the task environment was loaded and the participants solved 20 problems. Before starting the task, the participants were told that a component could be faulty more than once within the 20 trials. They were also told to move the mouse pointer to the components that they were attending and analyzing (cf. Freeman & Ambady, 2010; Mueller & Lockerd, 2001). After figuring out which component was faulty, the participant should select it with a mouse click. The first two faults within the 20 problems were either early faults (PS and EB1, or PS and EB2), as in Ritter and Bibby (2008). During the debriefing, the participants were thanked for their participation.

IV. Results

The user study results are based on the analysis of 35 out of 37 participant data sets. The two participants who were dropped had error rates in finding the fault (clicking on the broken component) of 45% and 55%; we do not know why their error rates were so high. The remaining subjects had an average error rate of 7.2%. The strategy models that we developed do not predict errors.

The data show that subjects' problem-solving times had different patterns across the same problem orders (with similar error rates). The response time for a trial consists of the time to solve the task, the time to move the mouse to the component, and the time to click the mouse button. The dexterity to move the mouse is considered equal across participants due to its

common use. The low average error rate indicates that the participants had equal abilities to solve the task, except the different pattern of response times to the same stimulus sets suggests that the participants developed different strategies.

Response time analysis

First, the 35 participants were analyzed together and their response times per trial were compared with the predicted response times per trial for the Diag-H model. The problem series the model saw corresponded to the series the participants saw. Fig. 4 shows that the predicted times and the actual response times per trial are highly correlated except for trials 1-2 which are always early in the circuit. To analyze the Diag-H model response time predictions, the predicted number of model cycles was used. We used linear regression to globally scale the Diag-H model cycle time to participant time. The mean intercept was 7.27 s and, therefore, is similar to the mean mouse movement time over all trials and participants with a mean mouse movement time of 6.25s. The additional time might represent visual encoding time. The best model cycle time was 57 ms per model cycle, which is within the range predicted by Newell (1990) of 30 to 300 ms. For each of the following analyses the predicted times (slope of model cycles * model cycles + intercept = 57 ms/mc * model cycles + 7.27s) are compared to the participant times. Diag-H's predictions on each series for each participant's 20 response times correlated $r^2 = .20$. This approach was used to calculate the models' problem-solving times in the remainder of the analyses for creating the graphs to show the relative performance for each strategy; the correlations reported use model cycles directly.

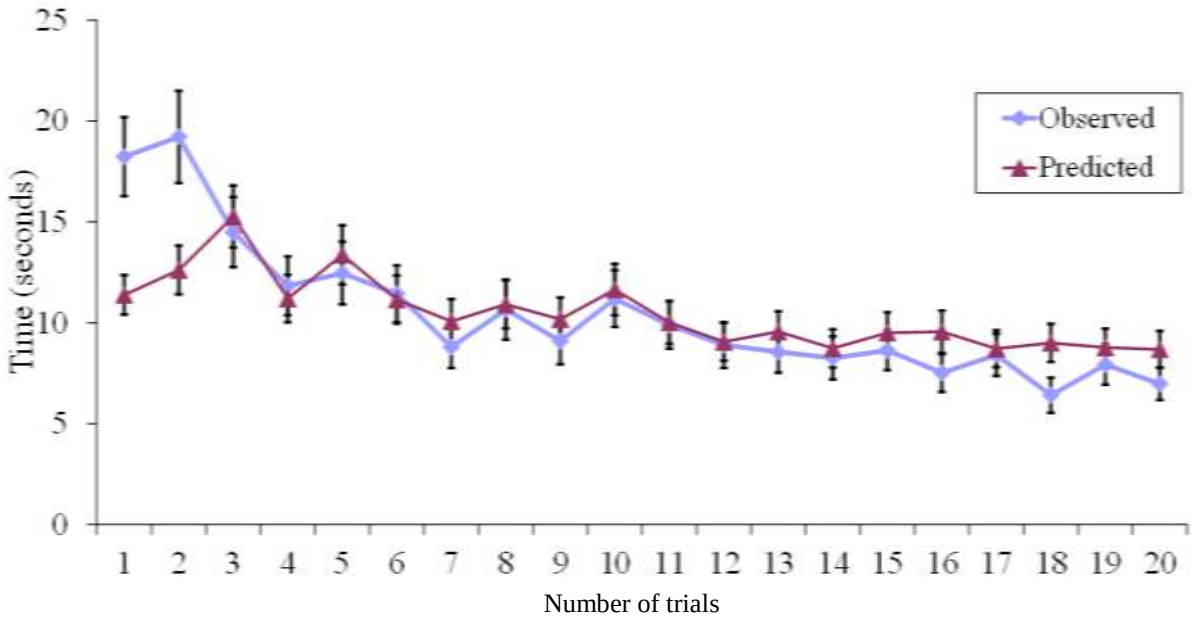


Figure 4. Predicted by Diag-H and observed problem-solving times (means and standard error) for trials 1 to 20 averaged over participants.

The predicted curve does not follow a power law for the first two trials because the series of problems start with two problems early in the circuit (starting from the power supply), what could be called easy problems. This was intentional to support initial learning, which appears to happen. The predicted times are also not smooth because not all possible series of problems were used. The remaining tasks are generally but not uniformly distributed across the range of problem difficulty. The first two observed trials shown in Fig. 4 differ from the predicted times more than the remainder of the trials. This effect of increased response time on the first two trials was not seen in the Ritter and Bibby (2008) data. This difference might be caused by the different instruction sets—the original Diag instructions gave a worked example while the new study instructions did not. The current participants performed the first two trials with less initial knowledge, adapting to the task and developing a strategy for the task. We also believe that the experimenter running the Ritter and Bibby (2008) study was directive in how to perform the task. Restricting the linear regression to trials 3 to 20 increases the correlation between predicted time and measured time, $r^2 = .23$. We categorized the first two trials as an orientation phase and

therefore as an initial part of the strategy developing process. The purpose of this work is not to show the time course of strategy development, but to show that different strategies are used. Because we did not model this phase in detail only response times of trials 3 to 20 will be examined further. This does suggest that revisions and extensions to the model should include more attention to the early proceduralization of knowledge, which the model currently assumes has been done.

The time to solve problems of four of the participants correlates well ($r^2 > .5$) with the initial strategy (Diag-H) time predictions. The four participants who used the Diag strategy supported the results of Ritter and Bibby (2008) because this correspondence while both model and participant are learning suggests that some participants use the Diag strategy to solve the Fault Finding Task. This also means that the remaining N = 31 (88%) participants do not fit well to the Diag-H time predictions, and, therefore, might have used different strategies or shifted between strategies. Other interpretations of the findings are discussed below. Thus, this dataset provides an excellent basis for finding new strategies.

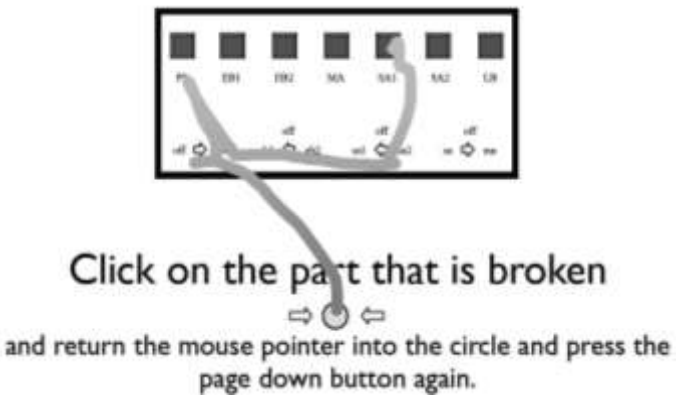
The lack of use of the Diag strategy can be explained by reference to the example problem given as part of the instructions in the original study. With the help of this example, the participants in the original study might have developed a strategy before performing the task. Importantly, they were given feedback concerning the correctness of their decision. Therefore, the participants could solve the task from the first trial. Because the new study instructions did not have a task example, the participants had to develop their own strategy for the task and were likely uncertain about their solutions because no feedback was given. This may explain the lower error rates in the other studies where feedback was given (Bibby & Payne, 1996; Ritter & Bibby, 2008).

V. Implementing and testing new Diag strategies

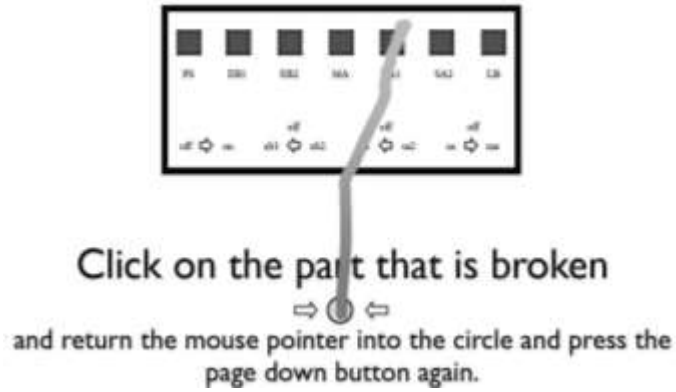
Participants were instructed to "If possible, move the mouse pointer always to the component you are thinking of at the moment." A reference point on every spacer slide was used to orientate the mouse movement as a reference point. The participants were told to return the mouse to this point before moving to the next task slide. This increased the value of the mouse movement data because the order in which participants attended the components was captured. Unfortunately, not all participants used the mouse as instructed.

The K-means algorithm (MacKay, 2003) was used to analyze the individual mouse movements to find potential clusters of strategy users. This approach does not provide definitive strategies but merely helps the researcher find some if they are there. For every participant several mouse move features were calculated: the mean number of mouse movements, the mean time for mouse movements, and the standard deviation of both. A mouse movement is defined as a new position of the mouse, gathered with a 200 Hz sampling rate. These values served as feature vectors (Table 2) for the K-means clustering algorithm. The K-means algorithm clusters N (the number of all different participants) objects based on the individual feature vectors into K (the number of result clusters) clusters with $K < N$. Considering the observations during the experiment, three different types of movement patterns could be identified, therefore clustering was started with three (K) initial clusters. Fig. 5 shows recorded mouse movement routes that support the results from the K-means algorithm.

Group (a) Clear mouse movements over components.



Group (b) Clicked directly on part



Group (c) No clear pattern

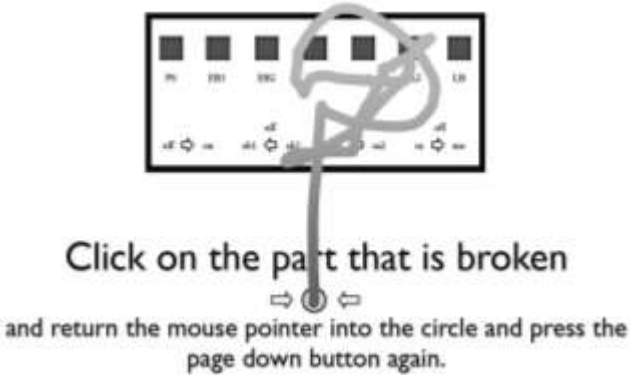


Figure 5. Example trials from every mouse movement group, solving fault SA1 in trial 3 of stimulus 11 (start point is dark grey and leads to endpoint in light gray; data points are from mouse movement logs).

Through the algorithm, the participants were assigned to the three groups a, b, and c shown in Table 2. The features of these three groups can be described as follows:

(a) Participants who used the mouse to follow a route with a clear movement pattern. Their movement routes can be described as their path of attention while solving the task. Their number of mouse movements was medium to low and the average mouse movement times were low. These could be used to infer new strategies.

(b) Participants who clicked directly on the faulty part. Their movement routes can be described as a straight line from the reference point to the component. Their number of mouse movements was low and the average mouse movement times were medium to high. These could not be used to infer new strategies.

(c) Participants who used the mouse with no clear movement pattern or did not return the mouse to the reference point. Their movement routes do not follow an identifiable order. Their number of mouse movements was high and the average mouse movement times were medium. These could not be used to infer new strategies.

Table 2. K-means results used to cluster the mouse movement behavior into three groups.

	Cluster		
	(a)	(b)	(c)
Participant Id's	A3_8	A2_9	A1_3
	A8_10	A9_7	A5_4
	A11_1	A10_6	A6_2
	A12_2	A13_3	A7_5
	A14_4	G1_7	G20_11
	A15_5	G2_8	
	A16_6	G4_10	
	G3_9	G6_2	
	G10_6	G7_3	
	G11_7	G8_4	
	G13_9	G9_5	
	G15_11	G12_8	
	G21_11	G14_10	
		G16_11	
		G17_11	
		G18_11	
		G19_11	
Total	13	17	5

In the next section the mouse moves in group (a) are used to generate hypotheses about the strategies the participants used. Due to learning, all participants changed their mouse movement behavior while solving the stimulus set. For example, participants from group (b) with practice moved even more directly to the answer than in their first trials, and some participants from group (a) switched to group (b) within their last trials. However, participants were assigned to those groups representing their prevailing mouse movement behavior. Further participants or more direct requirements to move the mouse would likely lead to further strategies being discovered.

Strategy identification was carried out with all 35 participants, even if they correlated well with the Diag-H strategy time predictions. We analyzed their mouse movement behavior to get insights into how they solved the task. The participants from mouse movement group (a) were the most useful. Participants from group (b) sometimes started with the behavior of tracing their

problem-solving steps, as in group (a), but switched to a direct and uninterpretable movements. Group (c) could not be used even if participants moved their mouse more during their last trials (trial > 15) because their answer times had gotten too fast to distinguish between strategies clearly.

The mouse movements of group (a) (noted in Table 2) were examined for strategy patterns. To identify strategies, the mouse movement data were analyzed by hand. The order in which components were attended was noted, for example, "PS," "EB1", and "SWITCH1". These combinations were retraced to see if they supported solving the Fault Finding Task. Strategies were identified based on often used interaction patterns in the mouse movement data. Identified patterns that could potently solve the Fault Finding Task were categorized by participants and identified strategy. Patterns that could not be used to solve the Fault Finding Task but showed similarities to already identified strategies were also categorized with that strategy. Thus, subgroups of participants with similar strategy patterns and a set of possible strategies were identified. Several strategies correlate well with individual participant's performances. The five new strategies identified that participants used to solve the Fault Finding Task are explained next, then they are tested by comparison to data. The explanation of each strategy is connected to diagrammatic knowledge (Fig. 1), circuit knowledge (Fig. 2), and the DIAG problem space (Fig. 3). All strategies are implemented in Herbal and are online available[5]. It has to be noted that the mouse movement path is used to infer the strategy—the times for each stimulus and the learning across stimuli are driven by the architecture and the strategies as hypotheses are tested by comparing their predictions to data.

[5] http://acs.ist.psu.edu/code/diag-H/

The Diagram Selection strategy

The diagram selection strategy (DiagSelect) was developed based on the mouse movements of participants A13_3, A15_5, G8_4, and G3_9. DiagSelect selects the components by using the circuit knowledge, especially the order of switches, and a check of the interface status. In this strategy, only the components that are on an active path through the device are selected for checking. Therefore, the test component problem space of DiagSelect has only the CHECK-LIT test. If the component is lit up, it is not faulty. Otherwise, the faulty component is found. If a component is not faulty, the circuit knowledge is used to find the next switch in the circuit. This procedure of component selection and component testing is repeated until the faulty component is found. Based on the SA1 example in Fig. 1, the order of components to check would be 8 (as first switch), 1 (to check if it is lit up), 9 (as second switch), 3 (to check if it is lit up), 10 (as third switch), and 5 (light is checked and component is identified as faulty).

CheckLitSwitch

The check lights and then switch strategy (CheckLitSwitch) was developed based on the sequence patterns of participants A1_3, A9_7, and A12_2. CheckLitSwitch solves the Fault-Finding Task with a separation between attending to the lights and switches. At first, the interface knowledge is used to select the right-most component that is lit up on the interface. The second step uses circuit knowledge to identify the next switch in the circuit. Its switch position is checked to determine the following component. The light of the selected component is checked and if it is not lit up, the faulty component is found. Based on the SA1 example in Fig. 1, the order of components to check would be 3 (lit up and most right), 10 (next switch in the circuit), and 5 (light is checked and the component is identified as faulty).

When developing the CheckLitSwitch strategy, it was necessary to determine if the participants had examined the last faulty component before clicking on it. Theoretically, they did

not have to because the last checked switch always points at the faulty component. Practically, the observation of participant A12_2 shows that, after checking the switch, the participant moved the mouse directly to the component but did not click right away. An interpretation for these observations is that the participants did check if the component was lit. CheckLitSwitch represents this behavior by performing a light test on the component.

Diag-H Random Selection strategy

The Diag-H random selection strategy (Diag-H-ranSel) is an adaptation of the Diag-H strategy and is based on sequence patterns of participants A14_4, G15_11, and A5_4. The strategy starts similarly to Diag-H by selecting a component and checking if it is faulty. The decision to check the diagrammatic or interface indicated component is made randomly. Based on Fig. 1, an example order could be 2 (as the first random component to check), 9 (as indicator that 2 is not the faulty component), 5 (as the second random component to check), 10 and 3 (as indicator that 5 is the faulty component). Each component is checked one by one and no component is selected twice for checking. Therefore, one to seven components may need to be checked with Diag-H-ranSel. The basic idea is that not all participants seem to have a clear understanding, especially in the first 10 trials, of what an efficient problem-solving strategy is. Therefore, the assumption is that some of the steps are made with less insight.

The Upstream State strategy

The upstream state strategy (UpState) is based on the mouse movements of participants A19_11, A13_9, A8_10, and of some participants from mouse movement group (c). The strategy always starts with the faulty component from the previous trial.

The first trial (the power supply, PS) is solved in the same way as the Diag-H strategy. Then the faulty component is remembered and used as the initial position for the following trial.

The strategy is divided into two steps. The first step involves identifying the direction to proceed because if the fault from the previous trial was nearer to PS than the current fault, the strategy has to work backwards. The direction is determined based on the position of the fault from the previous trial and the light that is lit up and farthest right on the interface. This indicates the direction in which to proceed. In the second step, the strategy tests the components in the determined direction until the faulty component is identified. Because the initial position is based on the faulty component from the previous trial, the strategy works best if the previous and current trials have the same fault. Based on the SA1 example in Fig. 1, the order of components to check would be 6 (if 6 was the previous fault), 3 (as determination for the direction), 10 (as first step backwards from 6), 5 (as next component to check), and 3 (as indicator that 5 is the faulty component).

The Lit Indicator strategy

The lit indicator strategy (LitIndicator) is also based on the separation of attending to the lights and switches. The strategy was developed based on the mouse movement data of participants A10_6, G16_11, and partially of the participants from mouse movement group (c). LitIndicator identifies the faulty component in two steps. First, the number of lit up lights is counted. If the number is zero or three, the faulty components can be identified right away as 1 or 7 (Fig. 1). If not, the second step is to use the switches (9 and 10) as an indicator for the faulty component. Based on the SA1 example in Fig. 1, the order of components to check would be 1 to 7 (for counting two lit up lights), 9 (to check the direction), and 10 (to identify 5 as a faulty component).

The six different strategies have different learning curves, represented by the model cycles that are needed to solve a specific problem. Fig. 6 shows the distinct learning curves for

each strategy for stimulus set 5 (similar curves arise for the other stimulus orders). The learning curves vary because each strategy learns different things from each problem and transfer to new problems varies based on the strategy and what knowledge is used. .

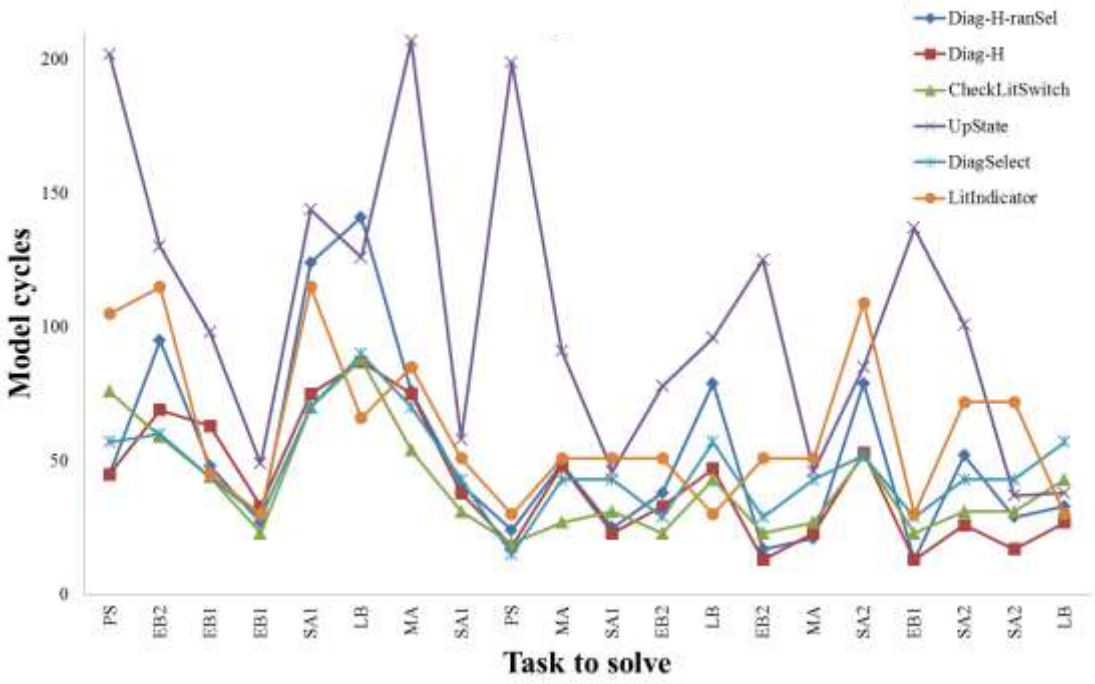


Figure 6. Number of model cycles per strategy for stimulus set 5.

Comparison of strategies to the participant data

The developed strategies were compared to the human performance data. The implemented models were used to solve the 11 stimulus sets. The results were six series of predictions (times to solve each of the 20 tasks) for each participant's stimulus set (one for each strategy: Diag-H, Diag-H-ranSel, CheckLitSwitch, UpState, DiagSelect, and LitIndicator). The prediction sets were compared individually to the participant data gathered from the additional study.

Table 3 shows the coefficients of determination (r^2) between the participants' solution times and each strategy's response times. The correlations per participant and strategy were filtered to identify the strategy with the highest correlation per participant. Using this method, all

35 participants were categorized into six strategy groups. Table 3 shows how well each strategy's predictions correlated with each participant's response times over the last 18 trials (trial 1 and 2 were removed due to the effects of the orientation phase). Participants who fit well to a strategy are defined by a correlation of $r^2 \geq .219$ (which is reliable, $p < .05$, $N=18$). In Table 3, 18 participants fulfill this condition.

Table 3. Participants, the strategies, and coefficients of determination per strategy and real data (r^2), sorted by best fit to strategy.
Bold values with * are the best fits for each participant. $r^2(18) > .219$ are reliable, $p<.05$
P next to a subject indicates that the best fit is plotted below in Fig. 7.

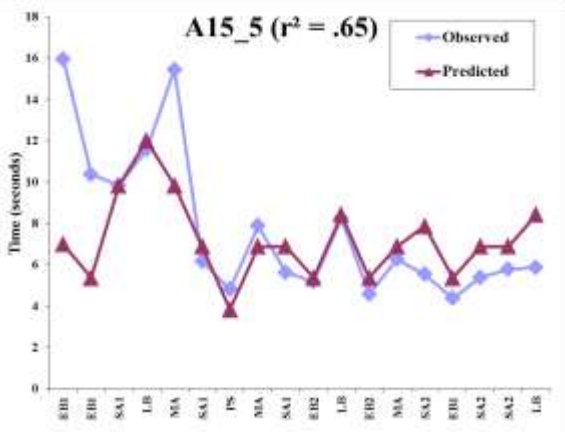
Participant	Diag-H	DiagSelect	CheckLitSwitch	Diag-H-ranSel	UpState	LitIndicator
A15_5[P]	**.65***	.46	.38	.39	.14	.10
A1_3	**.38***	.28	.32	.33	.00	.11
G1_7	.25	.16	.10	.21	.00	.08
A13_3	.23	.23	.19	.18	.02	.10
A8_10	.20	.03	.01	.08	.11	.12
G10_6	.17	.13	.09	.02	.02	.00
A11_1	.17	.12	.08	.06	.08	.00
G8_4[P]	.44	**.78***	.67	.49	.04	.01
G3_9[P]	.52	**.66***	.61	.64	.40	.09
A5_4[P]	.44	**.57***	.43	.41	.01	.01
G15_11	.16	**.32***	.16	.30	.16	.03
G18_11	.04	**.28***	.06	.06	.18	.02
G9_5	.24	**.28***	.27	.14	.05	.02
G6_2	.12	.17	.07	.02	.11	.04
G21_11	.10	.11	.04	.10	.03	.00
A9_7[P]	.44	.68	**.73***	.47	.04	.16
G17_11[P]	.33	.36	**.61***	.37	.07	.18
G14_10	.15	.33	**.33***	.26	.06	.11
G2_8	.24	.17	.26	.25	.00	.12
A16_6	.10	.15	.20	.16	.00	.09
A12_2[P]	.56	.58	.63	**.67***	.10	.45
A2_9[P]	.51	.53	.62	**.66***	.30	.37
A7_5[P]	.47	.49	.57	**.57***	.00	.09
G11_7	.10	.14	.05	**.34***	.05	.01
A14_4	.10	.13	.08	.15	.00	.04
A3_8	.09	.03	.03	.14	.00	.07
G19_11	.01	.22	.01	.00	**.38***	.01
G13_9	.24	.08	.12	.15	**.37***	.00
G4_10	.18	.01	.03	.03	.20	.02
G7_3	.03	.11	.07	.06	.14	.09
A10_6	.21	.12	.08	.14	.02	**.30***
G12_8	.13	.17	.19	.13	.05	.21
G16_11	.15	.07	.06	.10	.00	.19
G20_11	.00	.03	.01	.07	.00	.14
A6_2	.08	.00	.01	.07	.08	.09

Table 3 suggests that participants developed different strategies to solve the task. Some strategies are used more often than others, and some participants' behavior appears similar to a wider range of strategies. As shown in Figure 6, the strategies themselves have some similarity, and this will vary by the series of problems. This also explains why some participants' data match several strategies with high correlations at the same time. Some correlations, because they are reliable but smaller than found in Ritter and Bibby (2007), suggest that the participant may have used parts of multiple strategies. This fact could be caused by several circumstances, such as the knowledge of the participants, experience with similar tasks, or the ability to concentrate on the task. As in the Diag study, not every participant could be well predicted.

Participants who fit well to a strategy

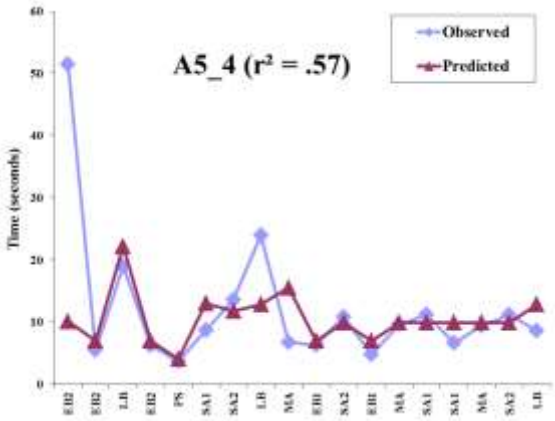
Fig. 7 shows the observed and predicted behavior for each of the participants who corresponded well to a strategy, showing how the predicted times match the observations. The plots show the correspondence over the series of trials while the model and the participants solved the same set of problems and learned. This supports the conclusion that participants chose different strategies to solve the Fault-Finding Task. Fig. 7 also shows that participants A15_5, A5_4, and A7_5 took longer to develop predictable strategies as their third trial took around five times longer than the predicted values. Because the performance of A5_4 and A7_5 fits to the predicted behavior for trials 4 to 20, they probably needed longer to develop their strategies. This indicates that the participants not only differ in strategy but also in the number of trials to develop a strategy. Only nine participants correspond well enough (correlation of $r^2 \geq 0.5$) to a strategy that the individual predictions indicate that they used only a single strategy.

Diag-H

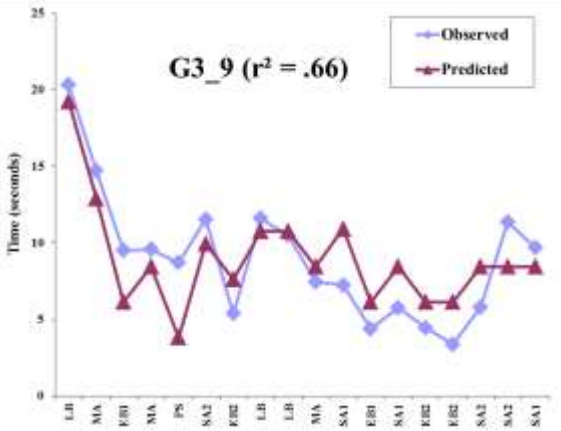


Task to solve

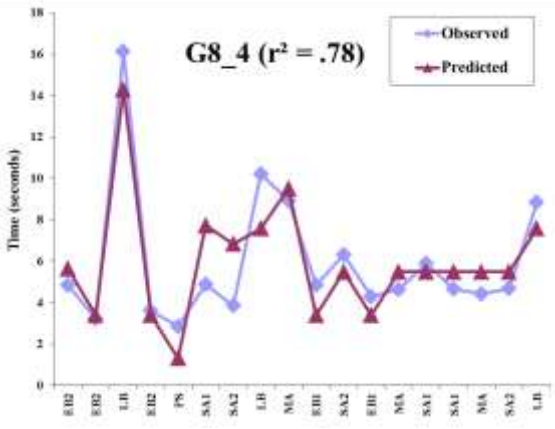
DiagSelect



Task to solve



Task to solve



Task to solve

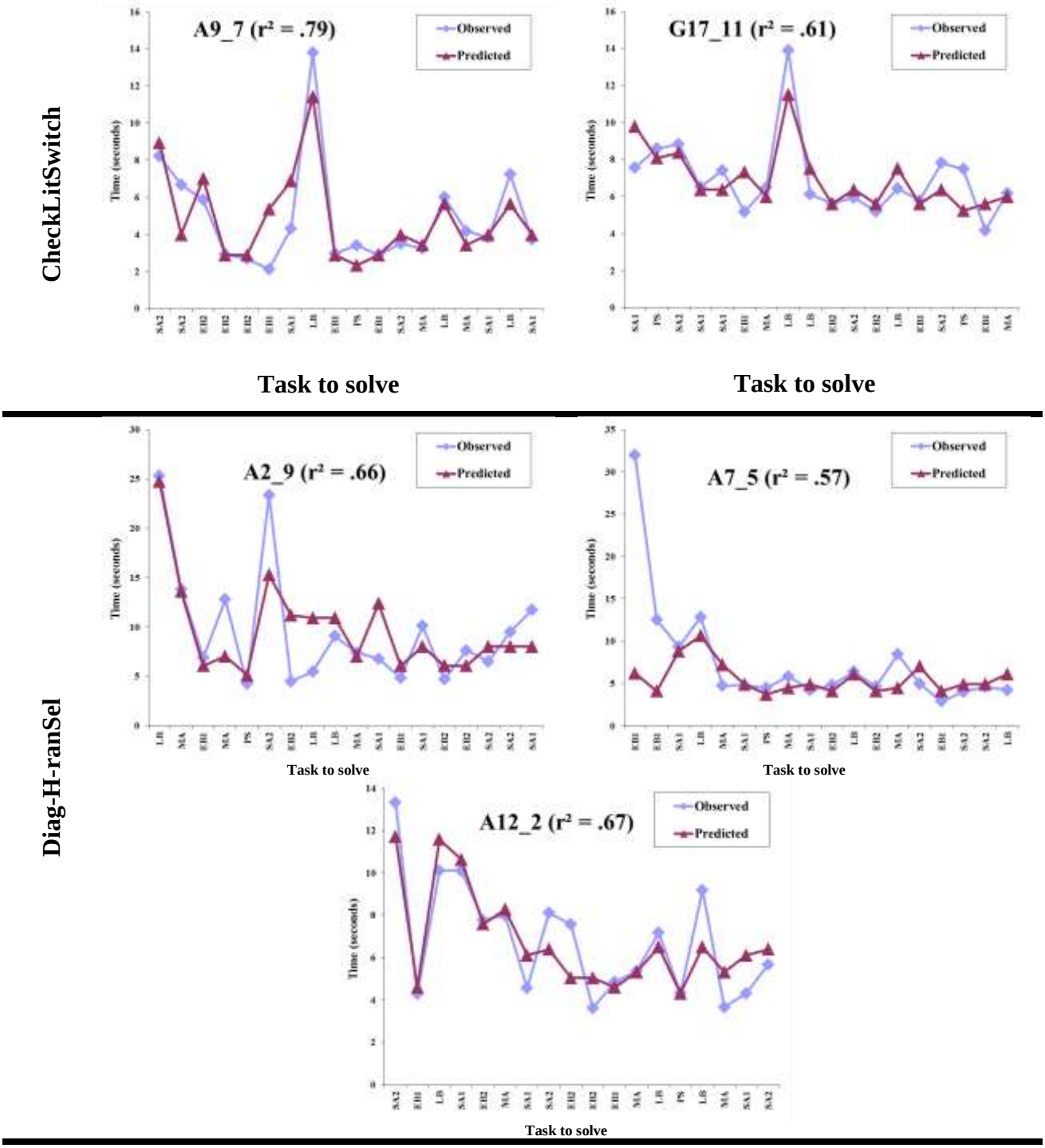


Figure 7. (continued) Predicted and observed problem solving times in task presentation order for participants with a correlation of $r^2 \geq 0.5$ for a single individual strategy. The predicted times are from the best fitting strategy. Series of problems are noted in the last part of participant id (e.g., _5), and scales vary based on observed time.

The participants' college majors were examined to see whether they had training in reading diagrams and schematics and whether their major and strategy choice correlated. Therefore, the participants were grouped based on academic major. The participants with a mechanical engineering major chose either DiagSelect or CheckLitSwitch with an equal probability. The other major groups appeared to use the strategies with an equal distribution.

VI. Summary of Comparisons

Five strategies for the Fault-Finding Task were developed based on the analysis of participant behavior and the mouse movement and response time clusters from an additional user study. The strategies were implemented in Herbal/Soar to create models to predict solving and learning the Fault-Finding Task. These models consist of reused (from Diag-H) and newly developed strategies. Every strategy model was run to predict the number of model cycles for the 11 stimulus sets. Human performance for these stimuli came from our additional study.

The participants were assigned to strategies based on their behavior correspondence to the predicted behavior from the strategies. The correspondence between data and model predictions showed that participants develop different strategies to solve the Fault-Finding Task. It was possible to fit 18 participants to individual strategies (r^2 between .26 and .98, $p < .01$).

The modeled strategies and the human participants learned over the task trials. We inferred 5 strategies from the participants early mouse movements. These were compared to the participant's response times over 18 trials while they both learned, with 18 participants' behavior well predicted (and no participant matching two strategies). 17 participant's response times were not matched by the original strategy (Diag) or the ones inferred from this set of participants. There remain further strategies it appears.

VII. Discussion and conclusions

In this article, we have developed a multiple strategies to explain participants' performance of a fault-finding task that illustrates the transfer of learning on a problem-by-problem level, and shows that the noisiness that most learning curves show can arise from differential transfer across different problems and different strategies. We first summarize these findings and then pull out lessons for understanding strategy use, and for understanding how strategies lead to variability in the learning curve. We finish by noting some limitations and future work.

These results and comparisons describe how behavior across individuals differs on a problem-solving task. We found different problem-solving strategies through examining human behavior. We implemented five strategies to solve the Fault-Finding Task, developed from the observed behavior of participants. So, whereas the early mouse move path is used to infer the strategy, the times for each stimulus and the learning across stimuli are driven by the architecture. Comparing the predicted and observed response times showed that participants used a wide range of strategies. Based on the correlation between human performance and model predictions, 18 participants (Table 3) could be assigned to at least one of the six strategies.

Humans evolve their strategies to improve their performance across similar problems (Siegler & Shrager, 1984). Implementing these strategies in a cognitive architecture can help to understand the development on a trial-by-trial basis. The models show and help understand the step-by-step learning process while solving the Fault-Finding Task. This analysis illustrates how strategies are used and improved during learning.

In this article, the Soar learning mechanism has proven itself capable of performing a task using models created with Herbal. The models' predictions matched the human behavior at a detailed level, including the rate of individual learning over a series of tasks.

Strategies within a task

The number of strategies in the relatively simple Fault-Finding Task is more than have typically been reported for a task before. Siegler (1987) found several strategies for arithmetic problems. Our results show that multiple strategies are used in a somewhat more complex task as well, a task that takes from 20 to 5 seconds to complete as learning occurs.

While some of these strategies appear to be similar, perhaps even in a trifling way, the results show that these strategies have different learning curves (shown in Fig. 6 and 7). The strategies also have different complexity, where some are simpler than others. This variety of strategies influences the process of learning while performing a task and how we can summarize how the task is performed and learned. The impact of the different strategies on individual performance times implies that even studies with simple tasks should be analyzed for the strategies that participants could use while solving the task. Thus, averaging over strategies seems particularly dangerous as tasks become more complex. If the strategies just do the same subtasks in a different order, it might be easier to ignore the effects of strategy. If the strategies differ in the information they use, or the order of interaction with a task (as in this task), then the effect of strategy will be more important.

Fault-finding strategies

We identified five new strategies for the Fault-Finding Task. These were used to predict human behavior. The strategies were developed based on a user study, but data analysis indicates that not all strategies used by subjects were identified. The multiple strategies were designed to model the aspects of learning and problem-solving in human behavior. The strategies always identified the faulty component with differing efforts, resulting in different numbers of sub-steps per strategy and therefore different learning curves.

The most frequently used strategies were those that participants could easily combine with their physical understanding (e.g., energy always flows from the PS to LB). Therefore, four out of the five strategies check the components forward through the circuit or from left to right on the interface. This could mean that the ability to develop new strategies is restricted by knowledge about the environment. To fully model a task and how it is performed, all possible strategies for solving the task have to be considered or at least as many strategies as can be identified. This is an elementary step for modeling human behavior, including learning and the variability in learning. Even though not all participants used multiple strategies to solve the task, all participants had to use at least one strategy. This result suggests that a range of strategies is more often used to perform a task, and that they will have to be understood and the strategies used have to be identified to more fully understand human behavior.

While the strategies arise from the order in which the participants examined the interface objects, the strategies' fits to the data series are not based on this order alone. The time to perform a strategy and the learning of a strategy within a series of problems did not come from the data but came from how the architecture implements and learns through repeatedly solving problems. Thus, while the strategy fit appears to be somewhat circular, the architecture provides the speed up due to learning and shows how time decreases with practice.

Variability in the learning curve due to strategy use

This work provides insights into the variability of learning, the transfer of learning, and the noisiness that most learning curves show. These predictions by the models suggest that the variability of learning may arise from several predictable sources. One source of variability is the differences in tasks when the tasks are examined in detail. The tasks in the stimulus sets vary in difficulty. Diag-H can be seen as a detailed implementation of Anderson's (Singley & Anderson, 1989) theory of transfer based on Thorndyke's identical elements theory (Thorndike &

Woodworth, 1901). Diag-H predicts that the different versions of a task when grouped on a high level are different in a sometimes subtle ways when viewed closely (e.g., the Fault-Finding Task for the MA or SA fault). In this case, the different faults vary in time to solve, the amount of learning, and potential for later transfer of learning.

A second source of variability is the differential transfer from not just the previous trial but from all previous trials. Diag-H suggests that this can be quite complicated, but can also be predicted through a detailed analysis.

A third source of variability, noted by other authors (Anderson, 2007; Siegler, 1988b), is the participants' differences in strategy choices—different strategies will give different learning curves over the same series of problems. The strategies also vary in how much what has been learned from previous problems helps with the current problem.

Finally, the results show that different strategies will have different learning curves and different transfers across problem sets. For example, strategies starting at PS (Fig. 1) are influenced by previous problems differently than strategies that start backward through the circuit. If a strategy works from left to right on the interface, a fault near the power supply will transfer little to the following trial where the laser bank is faulty. A strategy that works from right to left on the interface will transfer a lot from a fault near the power supply to the following trial where the laser bank is faulty. This is because it will see and use the whole circuit or interface to find the first fault and thereby train the complete task.

Keeping this differential transfer in mind, which our analysis did, provides another way to find out more about strategies and learning. Thus, some of the noise found in learning curves might not be unpredictable noise, just noise caused by our lack of knowledge of participants' strategy variants. The use of strategies and the poor fit for many participants suggests that further

strategies are being used or that the participants have the different speeds for subcomponents of the process (e.g., faster rule applier or slower working memory), which may appear as a different strategy.

Comparison of Diag to Diag-H

The comparison between the existing and new model brought interesting insights into how the models predict performance, why they predict similar results, and how we can build on Diag-H. These questions are important for further research on these models and especially for the research on strategies.

Diag (Ritter & Bibby, 2008) was used as a basis for comparison. The main difference is the percentage of participants who used the basic Diag strategy. Ritter and Bibby found that 80% of their participants matched the original Diag model, whereas in our study Diag-H matched only 4 of our 35 participants (11.4%) with a similarly high correlation ($r^2 > .50$).

The reasons why Ritter and Bibby (2008) predictions fit so well to the participant behavior might be explained by a different experimental setting. Their instructions were more directive and supported by an example of how to solve the task. This different set of instructions resulted in data with a clearer strategy development; it also demonstrates how influential seemingly small changes can be to a study design.

Despite the different populations and slightly different methods, the following findings are consonant. First, both studies show that participants used the Diag/Diag-H strategy and that the use of individual data and a cognitive model that learns helps to examine how cognitive skills speed up. Second, not all participants in the Diag study used the Diag strategy. Two participants in the Diag study did not fit the predictions. In the new study with less directive study materials, participants developed a range of different strategies to solve the Fault-Finding Task.

The model still has the problem that it does not explain how initial knowledge is acquired and how the participants know the order in which to apply the operators to solve the problem. We also had no measures of how well the subjects understood the task other than the performance data.

Further studies with the Fault-Finding Task should measure the initial understanding of the participants. This would enable a more comprehensive analysis depending on the individual participant's skills of understanding the Fault-Finding Task. It may even be possible to compare the developed strategy to how well the task was understood. The modeling and prediction of errors would provide insights into how to import knowledge of new strategies.

VIII. Limitations and future work

Although the fit of the multiple strategies in Diag-H to the data suggests that much of the variance in performance can be explained for many participants, the model may be improved in several ways. The fit between observed and predicted times suggests to us that even for the strategies with good predictions, there may be noise in the physical or mental realization of actions. Or there may be other not yet imagined sources of noise still to be accounted for. Also, we did not model individual differences in perception, motor control, the source of errors and subjects' potential for creating new strategies. Nor did we model differences in the creation of strategies or the switching between them, and these, too, could be important. There are also other behaviors that could influence performance on this task. The participants might have shifted between strategies. Finally, the model does not learn much after 20 trials, and while we do not have the data, it is likely that participants would continue to get faster after 20 trials. This suggests that further learning mechanisms, some in Soar already (Laird, 2012), and perhaps some yet to be added, will be needed to account for this learning regularity.

Using automatic or semi-automatic model development, such as genetic algorithms (GAs) (Kase et al., 2017; Lane & Gobet, 2005) or other types of machine learning (Best, 2013), could help generate cognitive models. These models and the strategy models in this article have a lot in common and can be extended. An automatic method for developing, testing, and comparing strategies to the observed data could be developed. By analyzing the strategy models, all of them were shown to include a process for selecting interface components and testing them. By dividing all strategies into subparts, it might be possible to identify only a few basic operators or triples of operators. An automatic process could reorganize these subparts to generate every strategy possible and compare their behavior to participant data. In this way, the optimal strategy and very likely alternative models for the observed data could be identified. The workload on the analyst of such a project could be kept low because the basic operators can be modeled in a high-level language and then stored. A program could build the model files, Herbal would compile them into models, and a GA would do the fitting.

It is interesting that only 33% of the participants who created their own strategies (in comparison to the 80% by Ritter and Bibby, 2008, who were more directed) could be covered by the predictions made by the new strategies. This shows that there is still a lot to learn about how humans develop and use strategies.

Acknowledgments

Support for this work was provided by ONR grant N00014-03-1-0248 and N00014-10-C-0281 / N091-086/P10008, and we thank Ute Schmid (several times), participants from Penn State and from Bamberg University, and the anonymous reviewers for comments and discussions. We thanks Steve Crocker, Moojan Ghafurian, Farnaz Tehranchi for comments on this paper.

IX. References

Altmann, E. M., & Trafton, J. G. (2002). Memory for goals: An activation-based model. *Cognitive Science, 26*, 39-83.

Anderson, J. R. (2007). How can the human mind exist in the physical universe?

Anzai, Y., & Simon, H. A. (1979). The Theory of Learning by Doing. *Psychological Review, 86*, 124-140.

Ashby, F. G., & Maddox, W. T. (2005). Human category learning. *Annual Review of Psychology, 56*.

Besnard, D. (2000). Expert error. The case of trouble-shooting in electronics. *Proceedings of the 19th international conference SafeComp2000*. Rotterdam, Netherlands.

Besnard, D., & Bastien-Toniazzo, M. (1999). Expert error in trouble-shooting: an exploratory study in electronics. *International Journal of Human-Computer Studies, 50*, 391-405.

Best, B. J. (2013). Inducing models of behavior from expert task performance in virtual environments. *Computational and Mathematical Organization Theory, 19*, 370-401. doi:10.1007/s10588-012-9136-8

Best, B. J., & Lovett, M. (2006). Inducing a cognitive model from examples provided by an optimal algorithm. *Proceedings of the Seventh International Conference on Cognitive Modeling*.

Bibby, P. A., & Payne, S. J. (1993). Internalisation and the use specificity of device knowledge. *Human-Computer Interaction, 8*, 25-56.

Bibby, P. A., & Payne, S. J. (1996). Instruction and practice in learning to use a device. *Cognitive Science, 20*, 539-578.

Card, S. K., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.

Cohen, M. A., Ritter, F. E., & Bhandarkar, D. (2007). *A Gentle Introduction to Herbal* (ACS 2007 - 1). Retrieved from acs.ist.psu.edu/herbal/herbal-tutorial-6-5-07.pdf

Cohen, M. A., Ritter, F. E., & Haynes, S. R. (2010). Applying software engineering to agent development. *AI Magazine, 31*, 25-44.

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science, 25*, 315-355.

Delaney, P. F., Reder, L. M., Staszewski, J. J., & Ritter, F. E. (1998). The strategy-specific nature of improvement: The Power Law applies by strategy within task. *Psychological science, 9*, 1-7.

Elio, R., & Scharf, P. B. (1990). Modeling novice-to-expert shifts in problem-solving strategy and knowledge organization. *Cognitive Science, 14*, 579-639.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science, 8*, 305-336.

Freeman, J. B., & Ambady, N. (2010). MouseTracker: Software for studying real-time mental processing using a computer mouse-tracking method. *Behavior Research Methods, 42*, 226-241.

Gobet, F., & Lane, P. C. R. (2007). An ordered chaos: How do order effects arise in a cognitive model? In F. E. Ritter, J. Nerb, T. M. O'Shea, & E. Lehtinen (Eds.), (pp. 107-118). New York, NY: Oxford.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science, 27*, 591-635. doi:10.1016/S0364-0213(03)00031-4

John, B. E., & Lallement, Y. (1997). Strategy use while learning to perform the Kanfer-Ackerman Air Traffic Controller Task. Mahwah, NJ: Erlbaum.

Kase, S. E., Ritter, F. E., Bennett, J. M., Klein, L. C., & Schoelles, M. (2017). Fitting a model to behavior reveals what changes cognitively when under stress and with caffeine. *Biologically Inspired Cognitive Architectures, 22*, 1-9.

Kieras, D. E., & Meyer, D. E. (2000). The role of cognitive task analysis in the application of predictive models of human performance. In J. M. C. Schraagen, S. E. Chipman, & V. L. Shalin (Eds.), *Cognitive task analysis*. Mahwah, NJ: Erlbaum.

Könik, T., O'Rorke, P., Shapiro, D., Choi, D., Nejati, N., & Langley, P. (2009). Skill transfer through goal-driven representation mapping. *Cognitive Systems Research, 10*(3), 270-285. doi:10.1016/j.cogsys.2008.09.008

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (2006). RUI—Recording User Input from interfaces under Windows and Mac OS X. *Behavior Research Methods, 38*, 656-659.

Laird, J. E. (2012). *The Soar Cognitive Architecture* (1 ed.). Cambridge, MA: MIT Press.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Lane, P. C. R., & Gobet, F. (2005). *Discovering predictive variables when evolving cognitive models. In Pattern Recognition and Data Mining.* Paper presented at the Third International Conference on Advances in Pattern Recognition, ICAPR 2005, Bath, UK, August 22-25, Heidelberg, Germany: Springer.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 311-334). Hillsdale, NJ: Erlbaum.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Models of competence in solving physics problems. *Cognitive Science, 4*, 317-345.

Lebiere, C., Wallach, D., & West, R. L. (2000). A memory based account of the prisoner's dilemma and other 2x2 games. In *Proceedings of International Conference on Cognitive Modeling* (pp. 185-193): Gronigen, NL: Universal Press.

Lehman, J. F., Laird, J. E., & Rosenbloom, P. (1996). A gentle introduction to Soar, an architecture for human cognition. In S. Sternberg & D. Scarborough (Eds.), *Invitation to cognitive science* (Vol. 4, pp. 211–253). Cambridge, MA: MIT Press.

Lesgold, A. M., Lajoie, S., Bunzon, M., & Eggan, E. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabay, & C. Scheftic (Eds.), *Computer assisted instruction and intelligent tutoring systems*. Hillsdale, NJ: Erlbaum.

MacKay, D. (2003). An example inference task: Clustering. *Information Theory, Inference and Learning Algorithms.*, 284-292.

Meyer, D. E., Kieras, D. E., Lauber, E., Schumacher, E., Glass, J. M., Zurbriggen, E. L., . . . Apfelblat, D. (1995). Adaptive executive control: Flexible multiple-task performance without pervasive immutable response-selection bottlenecks. *Acta Psychologica, 90*, 163-190.

Mueller, F., & Lockerd, A. (2001). Cheese: Tracking mouse movement activity on websites, a tool for user modeling. In *CHI'01 extended abstracts on Human factors in computing systems* (pp. 279-280): ACM.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1991). Formulating the problem space computational model. In R. F. Rashid (Ed.), *Carnegie Mellon Computer Science: A 25-Year commemorative* (pp. 255-293). Reading, MA: ACM-Press (Addison-Wesley).

O'Reilly, R. C., & Munakata, Y. (2000). Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain.

Ohlsson, S. (2007). Constraining order: Order effects in constraint-based skill acquisition. In F. E. Ritter, J. Nerb, E. Lehtinen, & T. O'Shea (Eds.), *In order to learn: How the sequences of topics affect learning* (pp. 151-165). New York, NY: Oxford University Press.

Pavlik, P. I. (2007). Timing is in order: Modeling order effects in the learning of information. In F. E. Ritter, J. Nerb, E. Lehtinen, & T. O'Shea (Eds.), *In order to learn: How the sequences of topics affect learning* (pp. 137-150). New York, NY: Oxford University Press.

Ritter, F. E., & Bibby, P. A. (2008). Modeling how, when, and what is learned in a simple fault-finding task. *Cognitive Science, 32*, 862-892.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (1987). Meta-Level Architectures and Reflection. 227-240.

Seibel, R. (1963). Discrimination reaction time for a 1,023-alternative task. *Journal of Experimental Psychology, 66*, 215-226.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology, 115*, 250-264.

Siegler, R. S. (1988a). Individual differences in strategy choices: Good students, not-so-good students, and perfectionists. *Child Development, 59*, 1-38.

Siegler, R. S. (1988b). Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General, 117*, 258-275.

Siegler, R. S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? *Origins of cognitive skills*, 229-293.

Simon, H. A., & Reed, S. K. (1976). Modeling strategy shifts in a problem-solving task. *Cognitive Psychology, 8*, 86-97.

Singley, M. K., & Anderson, J. R. (1989). *Transfer of cognitive skills*. Cambridge, MA: Harvard University Press.

Thorndike, E. L., & Woodworth, R. S. (1901). The influence of improvement in one mental function upon the efficiency of other functions. Part I. *Psychological Review, 8*, 247-261.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science, 15*, 1-47.

VanLehn, K., & Jones, R. M. (1993). Learning by explaining examples to oneself: A computational model. In S. Chipman & A. L. Meyrowitz (Eds.), *Foundations of knowledge acquisition: Cognitive models of complex learning* (pp. 25-82). Boston, MA: Kluwer.

White, B. Y., & Frederiksen, J. R. (1987). Qualitative models and intelligent learning environments. In R. W. Lawler & M. Yazdani (Eds.), *Artificial intelligence and education: Vol. 1. Learning environments and tutoring systems* (pp. 281-305). Norwood, NJ: Ablex.